Status of the IPbus Project

Robert Frazier

Greg Iles, Marc Magrans, Dave Newbold,
Andrew Rose, Dave Sankey, Tom Williams

Institutes: Bristol, Imperial, RAL, CERN

Talk Overview

- IPbus concept recap
- The last 6 months
- *μHAL status*
- *ControlHub* status + performance
- IPbus Firmware status
- Address assignment solutions
- IPMI comments
- Future plans

IPbus is...:

- **A system for control of µTCA-based hardware over Ethernet**
  - Replacing the VME HAL, and associated CAEN drivers/controllers

- **Baseline system consists of:**
  - *IPbus Firmware*: receives and acts upon IPbus protocol instructions sent over UDP.
  - *µHAL*: the end-user programming interface.
  - *ControlHub*: forms a single point of contact with the hardware, serialising transaction requests from multiple active *µHAL* clients.

- **Developed entirely in-house**
  - Based on commonplace standards – no industry "lock-in".

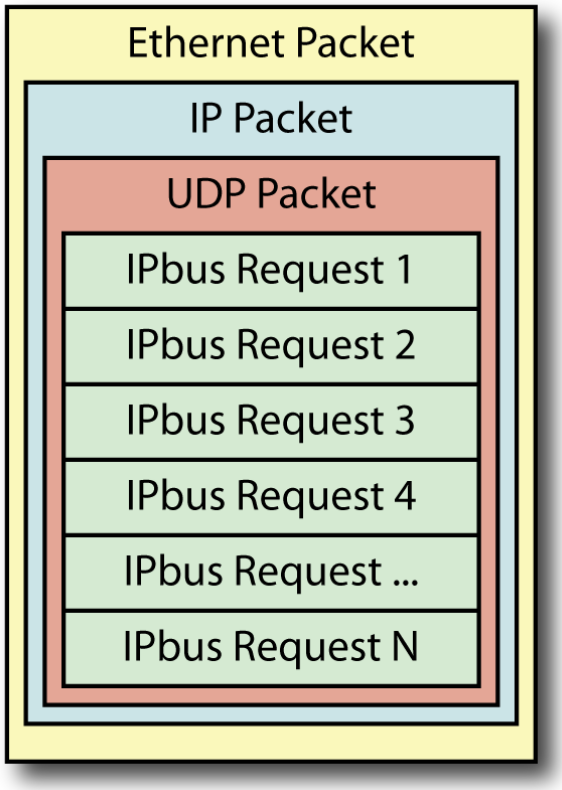
IPbus Protocol basics 1

- **The protocol describes the basic transactions:**
  - Read
  - Write
  - Non-incrementing Read/Write
  - Atomic Masked Write (Read/Modify/Write)

- **A32/D32**
  - Word-addressable, not byte-addressable
  - 16 GiB maximum addressable space per AMC

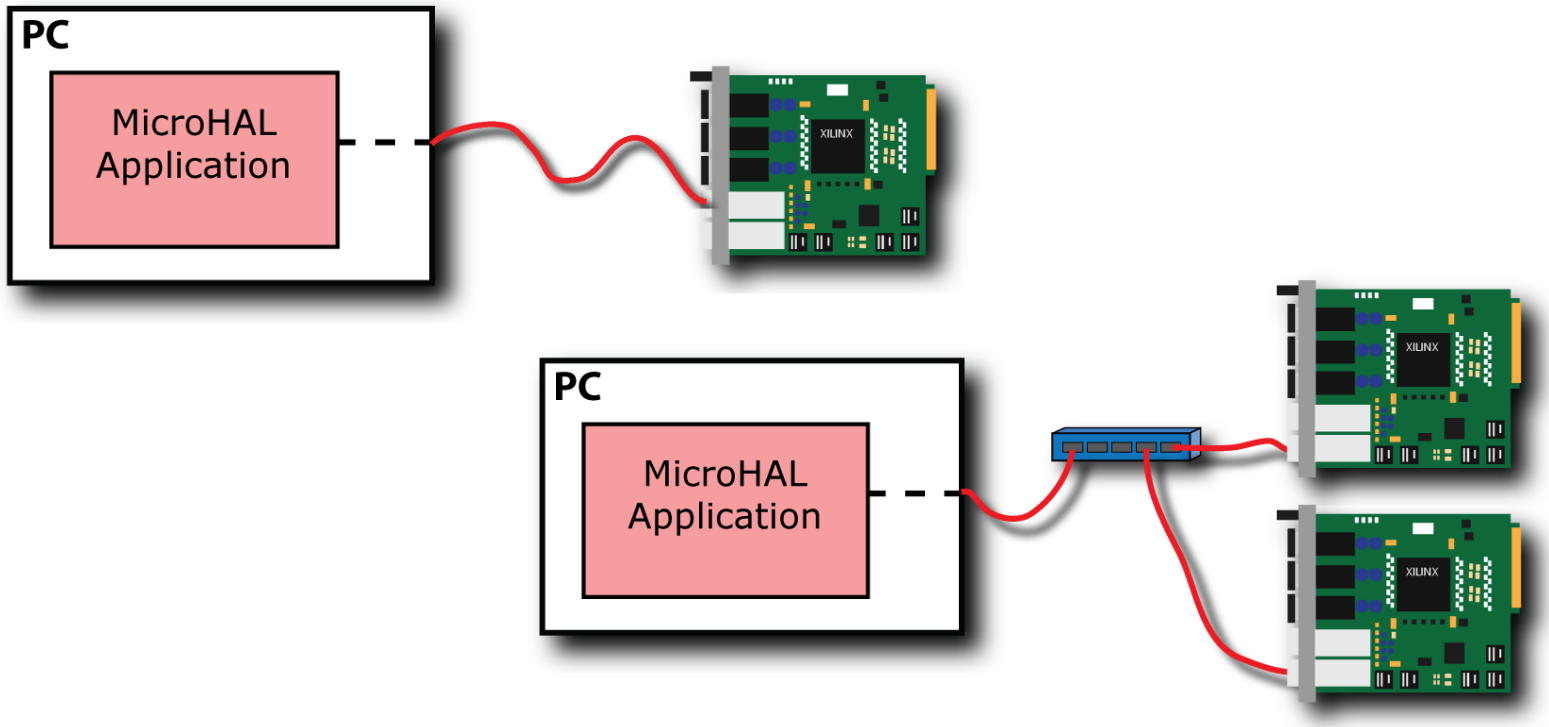
IPbus Protocol basics 2

- **Each transaction request/response is self-contained**
  - Has its own header and body
  - Transport protocol agnostic
- **Transactions can be concatenated together into the same packet**
  - Queue requests and dispatch when necessary
  - Improves network transport efficiency
  - Major difference to VME!



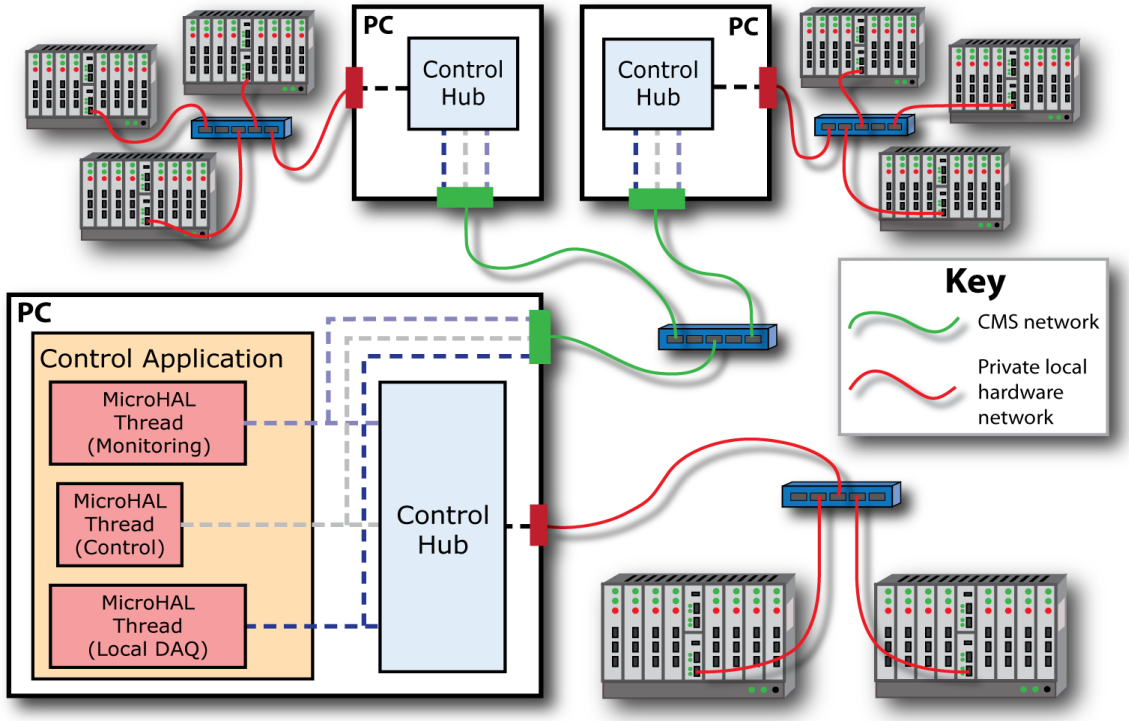
IPbus scalability

- **IPbus is designed to be very easily scalable:**
  - From simple bench-top testing of a single/few cards:



IPbus scalability

- **All the way up to something much, much bigger:**



Many topologies possible – this is just an example. More on this later.

The last 6 months

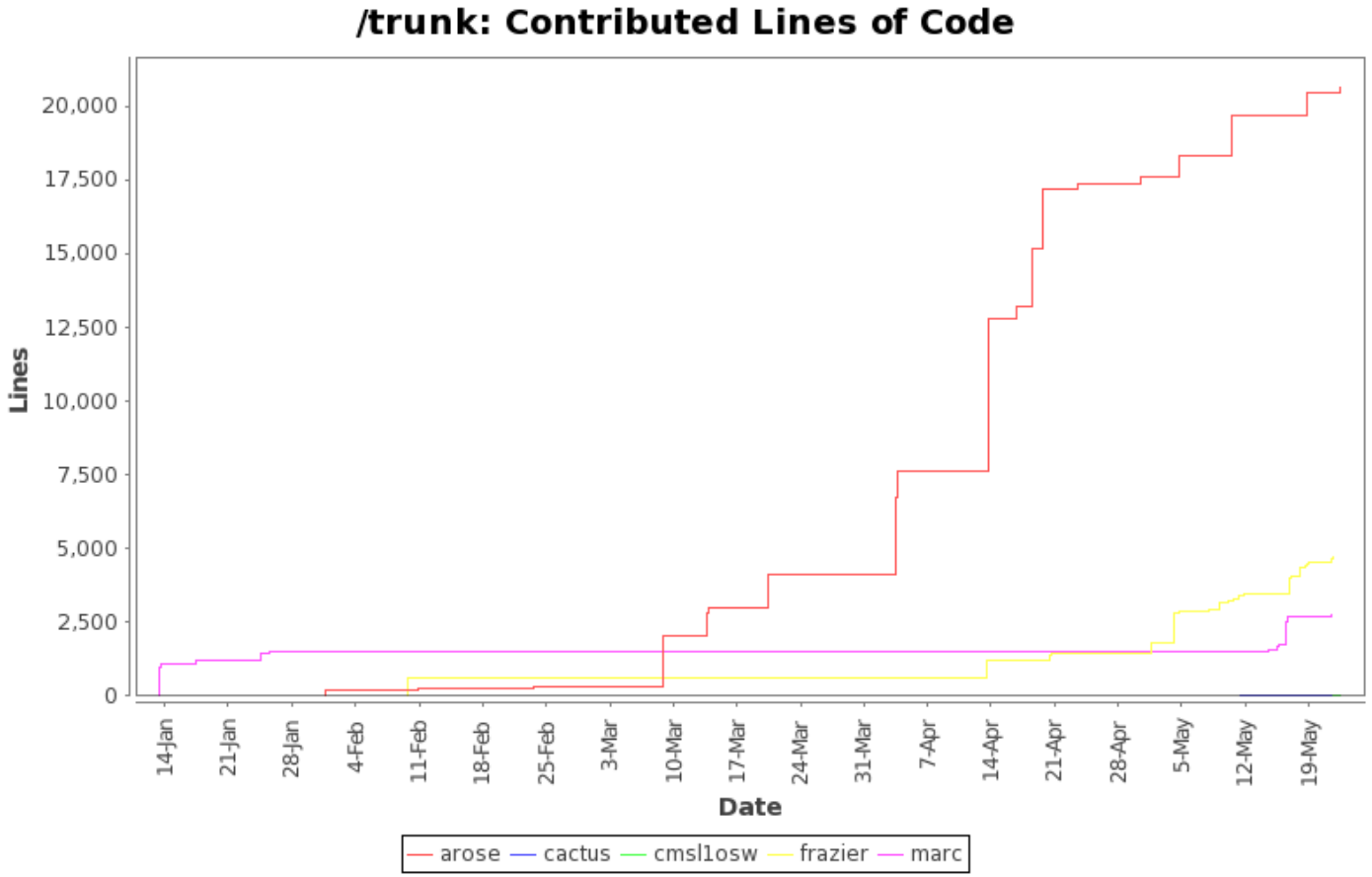
- **Many different groups developing hardware control technologies**
- **Marc Magrans becomes very worried…**
  - One project is better - or at least unite under common API
  - API review and requirements gathering begins at end of 2011
- **Online Co-ordination reviews competing systems Jan/Feb 2012**
  - Detailed presentation of the *IPbus* concept was given
    - https://indico.cern.ch/getFile.py/access?contribId=0&resId=0&materialId=slides&confId=164330
- **Online co-ordination gave ruling in early March**
  - *IPbus* will be hardware access baseline
  - *μHAL* to be reworked to form the common API, with *IPbus* as one of the protocol options.
  - *IPbus*/*μHAL* project to be integrated into CMS online software

The last 6 months

- **CMS requirements capture for:**
  - Network addressing + hardware address assignment
  - Locking and synchronisation
- **This is still ongoing, as much complexity here**
  - Particularly with regards to addressing
  - See Dave Newbold's section of this talk for more detail on this.

- **Mid/late March → now:**
  - IPbus project migrated into CERN-held SVN repository.
  - Work begins on required changes to *μHAL* & *ControlHub* software
  - Packaging of dependencies, integration into nightly build system, RPM build system, etc.

The last 6 months



Status today: *μHAL*

- **Supports IPbus Protocol v1.3 – direct or via *Control Hub***
- **Baseline features specified by API review complete…:**
  - http://cactus.web.cern.ch/cactus/documents/uhal_api_poposal/html/annotated.html
  - …but still in final stages of integration testing with the *Control Hub*
- **New config file:**
  - Abstract board names (e.g. `hcal.crate1.slot1`) to addresses
- **Some HCAL addon requests still in progress**
- **RPM packaging of externals and *μHAL* itself is largely complete**

- **Caveat: although C++ API is now fixed, the address-mapping config files will probably still evolve (see Dave's slides).**

Status today: *Control Hub*

- **New Control Hub is functionally complete + fully tested**
  - Still implemented in Erlang
  - 100% configuration-free implementation - just start it running
  - Compatible with IPbus Protocol v1.3

- **Still to do:**
  - Build RPMs of ControlHub and dependencies.
  - Some non-critical secondary features still need to be added:
    - Proper logging
    - Daemon startup script

Control Hub Scaling Test 1

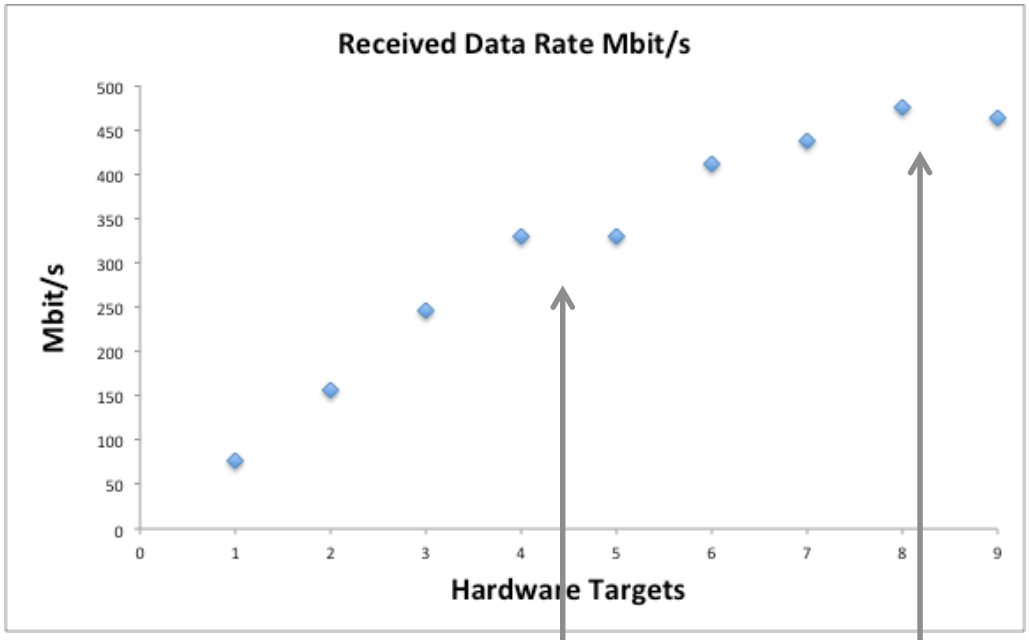
- **Many iterations of client sending 4 KB non-incr. read request to various numbers of hardware targets.**

- **Each client request for a particular target results in 30 UDP packets being sent (total read of 4 KB per iteration).**

- **Simple test-client used (not *μHAL*)**
  - Sends hand-crafted packet,
  - Waits for response, checks ok
  - Repeat.

- **Bandwidth results are for application bandwidth**
  - Only the returned useful payload in used in calculations.

Control Hub Scaling Test 1



Targets	Scaling cf. single Target
1	1.00
2	1.02
3	1.08
4	1.08
5	0.87
6	0.90
7	0.82
8	0.78
9	0.68

Core power/throttling state affects scaling?

Erlang scheduler starts using 2 cores

Erlang scheduler starts using 3 cores

Control Hub Scaling Test 2

- **Multiple clients sending requests for all 9 targets simultaneously.**
- **Sum the data rate received by *n* active clients, where n = 1 - 4**

Can't explain this scaling fully yet, but it's good!

Firmware Status

- Firmware components
  - IPbus control logic and fabric + Example slaves
  - UDP-driven and SPI-driven bus masters
  - Example Ethernet and clocking implementations
- Status
  - UDP firmware is now rather mature and debugged (50+ users)
  - Latest (final?) protocol changes in implementation and testing
  - Emphasis turning to performance and reliability enhancements
- Performance
  - Current firmware targets minimal resource usage
    - Can be used even in small low-cost FPGAs
  - Performance was so far not the key target
  - Ethernet performance strongly depends on buffer size
  - Buffer sizes are now parameterised to allow size / performance tradeoff
    - With a 16 jumbo frame buffer, we should reach gigabit wire speed

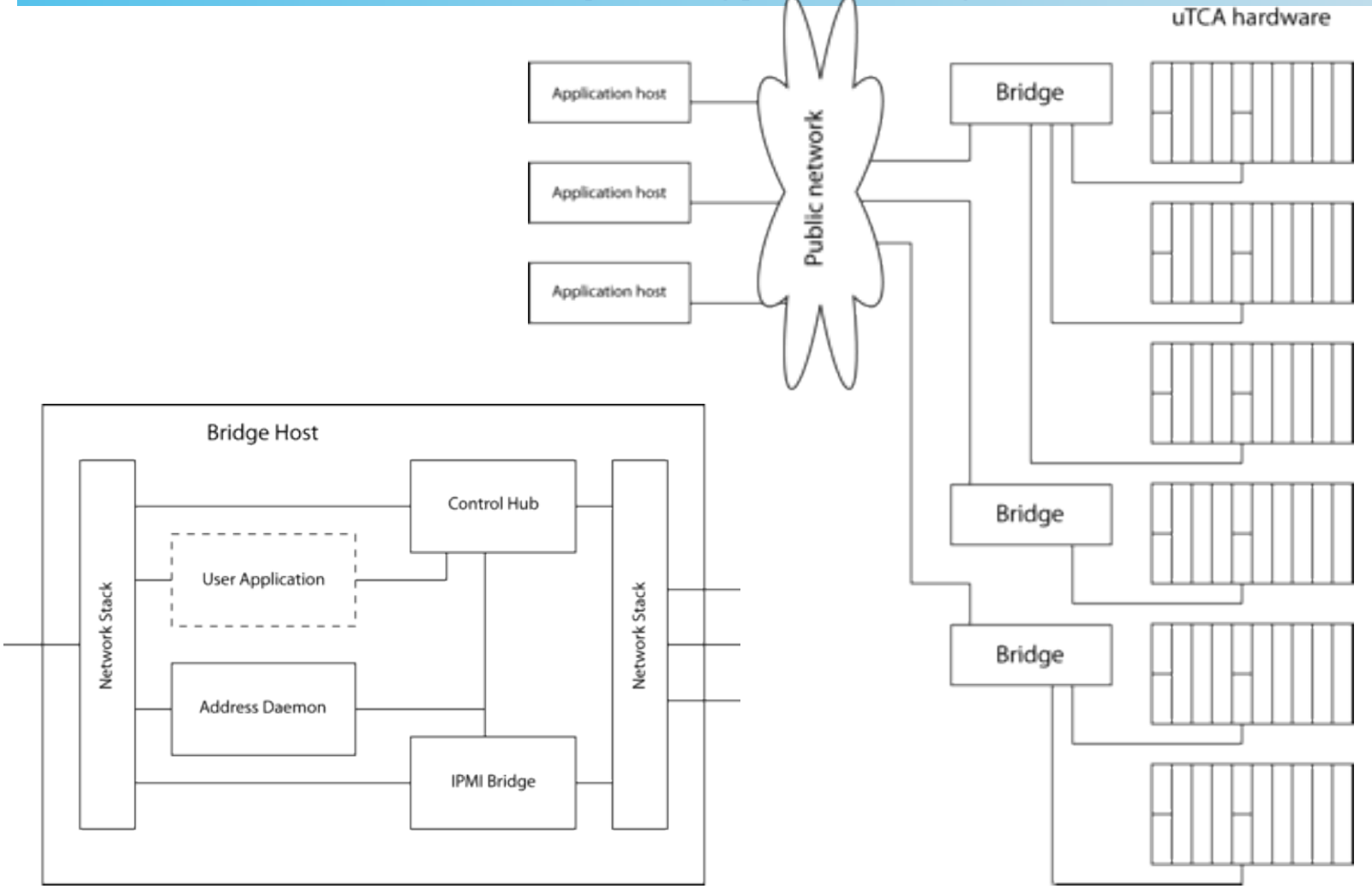
Addressing Scheme, etc

- Action from last meeting
  - Deliver a short document proposing an IP addressing scheme
  - Draft document may be found at: http://cern.ch/frazier/uHAL_network_addressing.pdf
  - Comments and input are necessary and welcome
- Covered in the (not-so-short) document
  - Network topology for connection of uTCA hardware
  - Assignment of MAC and IP addresses to hardware
  - Configuration of module addresses at startup / hot-swap
  - Mapping of 'symbolic' (uHAL) module names to addresses
  - Several questions and options – input needed
- Use cases considered
  - A full 'CMS counting room' setup with hundreds of modules
  - A test-beam setup with a single crate
  - Single module on a bench

Network & Addressing: Key Points

- Network topology
  - AMC modules are not connected to the general-purpose network
    - Packets cannot be directly routed from arbitrary machines to AMCs
  - Hardware is connected via a bridge machine ('firewall')
    - This can host a control hub, uTCA System Manager, DCS-IPMI bridge user application, etc
    - Nothing prevents a hardware control application to run directly here – the 'old model'
  - A single bridge machine can host several uTCA carriers
    - One interface on general-purpose network, *n* others connect to uTCA carriers
- Network addresses
  - All Ethernet-connected hardware has 'official' unique MAC address
  - MAC addresses are hardcoded or in non-volatile storage
  - IP addresses are assigned by an address assignment daemon
    - The daemon effectively forms part of the System Manager component
  - Two IP address assignment mechanisms: IPMI-based or RARP-based
    - The two mechanisms cover different system scales and use cases
  - IP addresses are from the 192.168.x.y private space

Network Topology / Bridge Host



Name – Address Mapping

- uHAL symbolic names
  - Hierarchical names of the type subsystem.crate.module
    - e.g. HCAL.sector2.module_a / GCT.input_crate.slot2
  - Naming scheme assumes nothing about the protocol or hardware
  - 'Slot' has no meaning at this level – but modules are numbered
- Mapping to IP addresses
  - Two-file configuration solution
  - Map subsystem.crate to a physical uTCA carrier (and a bridge host)
  - Map module name to a 'module number', used as last byte of IP address
  - The uHAL knows which bridge host to talk to
    - Completely transparent to user application – no knowledge of IP address is needed
  - The bridge host routing table sends packets to the right interface
- Future ideas
  - This scheme is not IP or UDP specific
  - Can extend to PCIe-connected modules (or even VME)

IPMI Comments

- Our (unambitious) goal
  - Find out how to (easily) obtain list of active AMCs via IPMI
  - Find out how to send arbitrary commands to AMCs
  - Test IP address assignment mechanism
- The experience
  - Not good: Open source tools do not 'understand' TCA IPMB topology
  - Sending raw commands is possible, and works, but ugly
    - Things will unravel fast if errors or lost packets occur on the IPMB
  - Vendors do not appear interested in support, fixing bugs
    - Possible to crash NAT MCH with malformed IPMI packets
- MMC code
  - IPMI address assignment needs agreed IPMI extensions by MMC
  - We have converged on an SPI interface between MMC and FPGA
  - Proposal: use SPI fns (Tom Gorski) for out-of-band access to IPbus space
    - This can include read-only config information and read-write configuration space

Future plans

- **Expect a first RPM release in late June**
  - Usable compiled from source before then if desperate.
- **Frequent and regular releases with new features after this**
  - Much initial effort gone into test, packaging + release machinery.
  - This effort is ~one-off
- **New features to come are:**
  - HCAL “addons”.
  - Support for new, higher-performance firmware
  - Locking + sychronisation features
  - Evolution of addressing

Conclusions

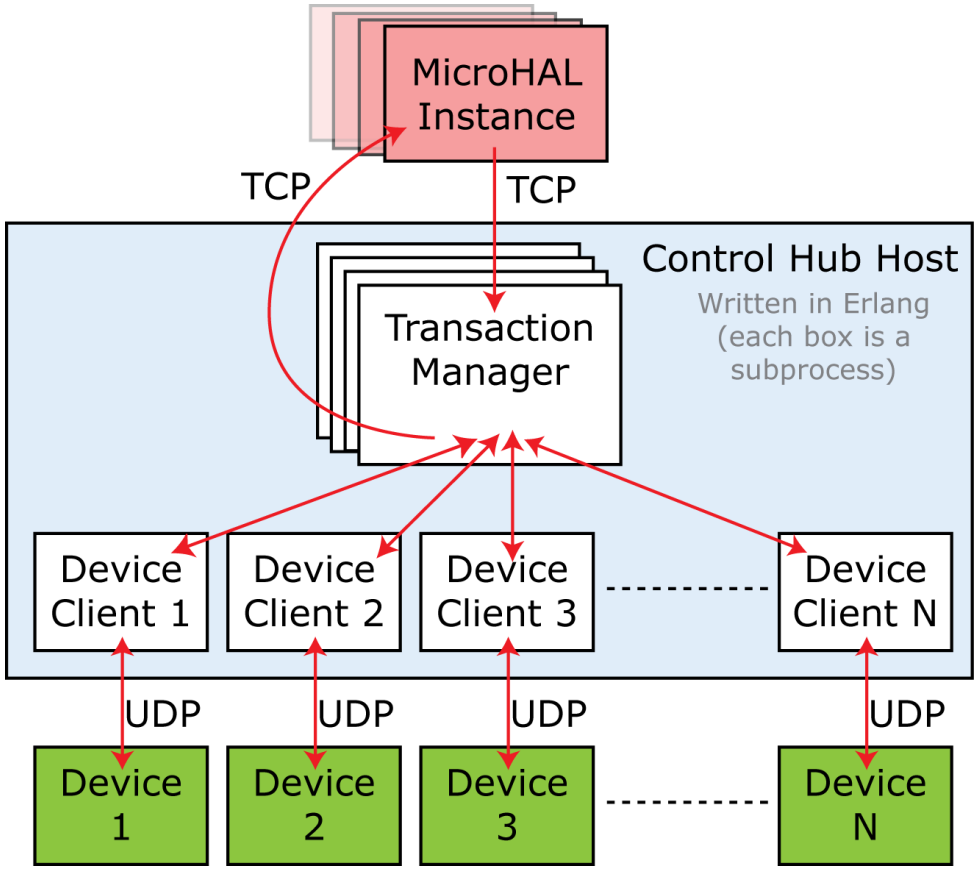
- **First RPM release of new *μHAL* and *Control Hub* expected in June**
  - Initial release will support small footprint IPbus v1.3 firmware only
- **Many of the more advanced requirements for a full-scale CMS deployment now understood**
  - Draft document now available for hardware address assignment
  - Needs further collaboration input and review
- **More needed…**

Backup

Software: Control Hub 1

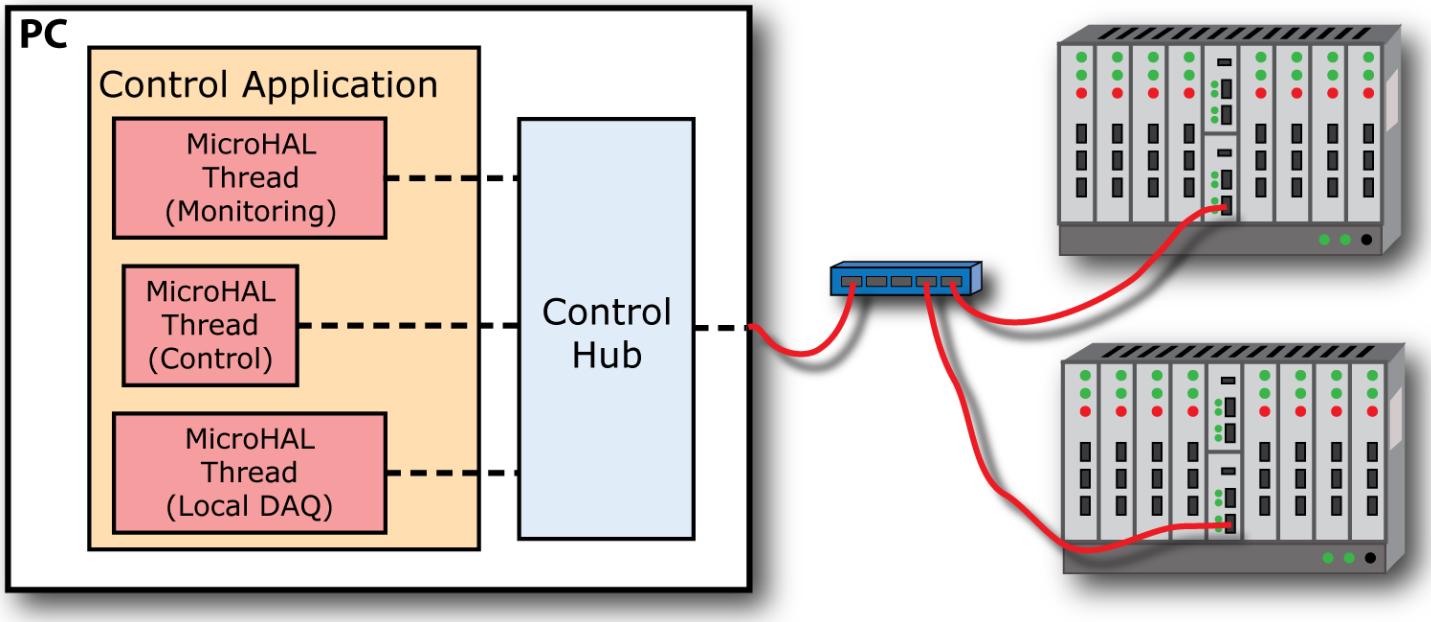
- **Analogous to a VME crate controller + driver software**
  - The Control Hub forms a single point of contact with the hardware
  - “Owns” the hardware network
  - Control Hub can also be thought of as an IPbus packet router
- **Current functionality**
  - Accepts up to 64 simultaneous microHAL clients
  - Makes best use of available bandwidth
    - Bandwidth to each board is multiplexed across gigabit connection
  - Highly scalable – can control multiple crates of boards
    - Pretty much only limited by number of Ethernet connections and CPU cores you can cram in a single rack PC.

Software: Control Hub 2



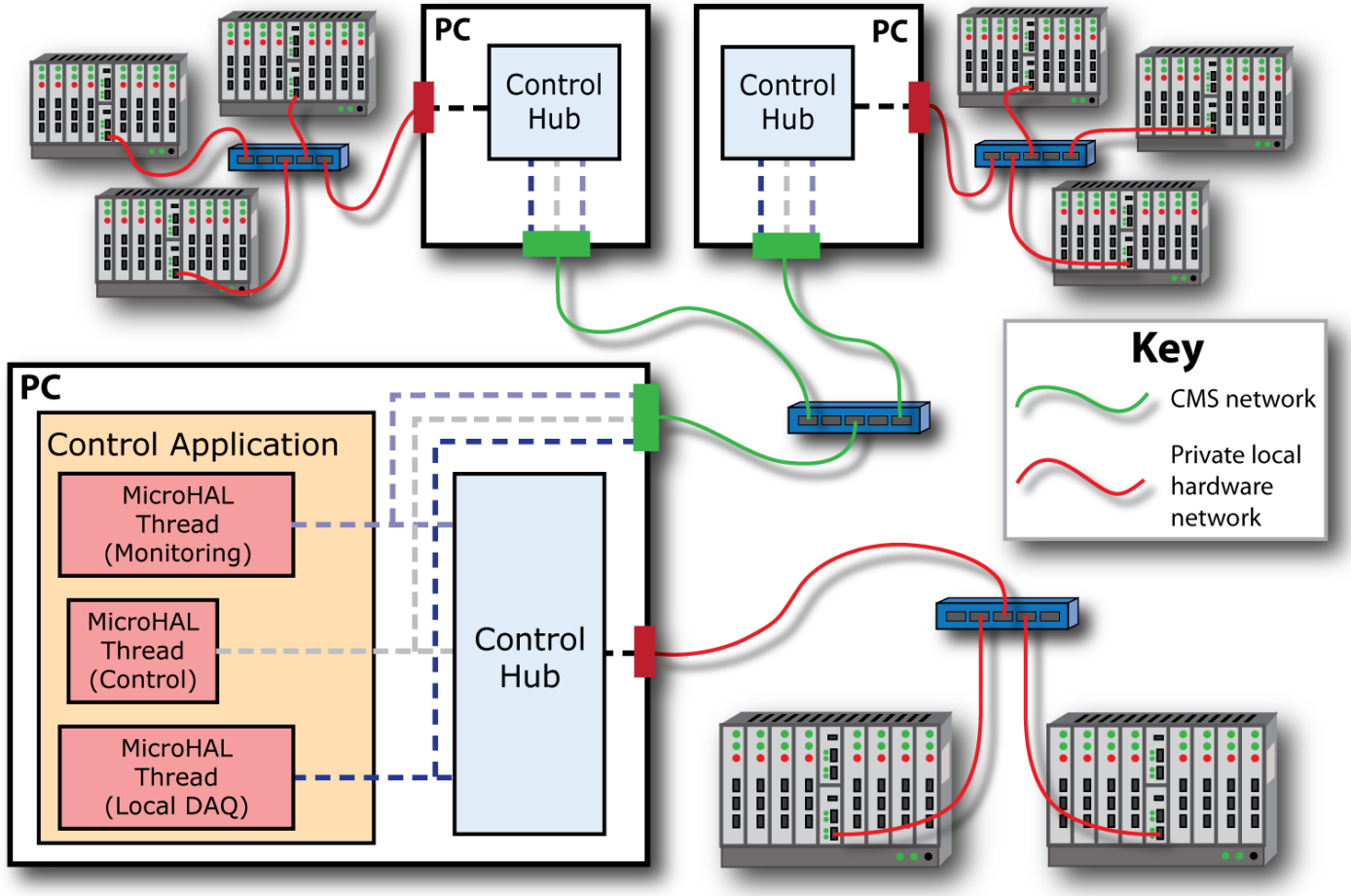
Use-cases: test-beam 2

- "Control Hub" acts as single point of contact with hardware



- Control app. can now safely instantiate independent client threads

Use-cases: full-scale 2



Testing IPbus 1

- **Extensive test system in Bristol**
- **On three rack PCs, we run:**
  - Multiple μHAL instances
  - Single Control Hub
- **Many (logical) IPbus hosts…**
  - Running on five physical development boards
    - 1 x SP605
    - 3 x Atlys SP605 Equiv.
    - 1 x Avnet V5

Testing IPbus 2

- **And at Imperial we have**
  - The TMT calorimeter-trigger demonstrator!
    - 6 x Mini-T V5 boards
- **IPbus over UDP has so far proved very solid**
  - Extensive soak-testing performed
  - Good understanding of packet lost rates (1 in ~200 million)